

# A Component Model of Spatial Locality

---

Xiaoming Gu  
Intel China Research Center  
Beijing, China

Joint work with Ian Christopher,  
Tongxin Bai, Chengliang Zhang  
and Chen Ding



UNIVERSITY *of*  
ROCHESTER

# Memory Wall and Locality

---

- Memory wall
- Locality
  - temporal locality: enable cache reuse
  - spatial locality: implicit data prefetching and better cache utilization

# Spatial Locality

---

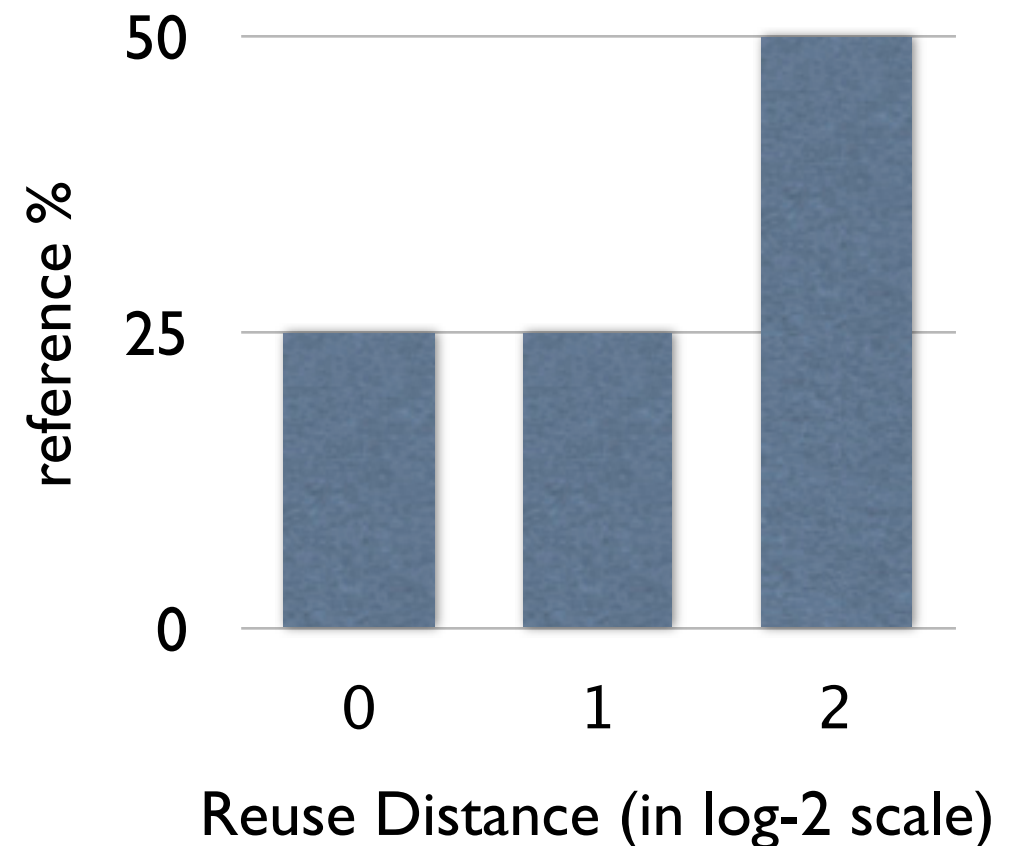
- Three types
  - intra-block spatial locality ★
  - inter-block spatial locality
  - adjacent-block spatial locality ★

# Reuse Distance

- Reuse distance
  - the volume of data referenced between this and the previous access
- Reuse signature
  - the distribution of all finite reuse distances

access trace	A	B	C	A	A	C	B
rd	$\infty$	$\infty$	$\infty$	2	0	1	2

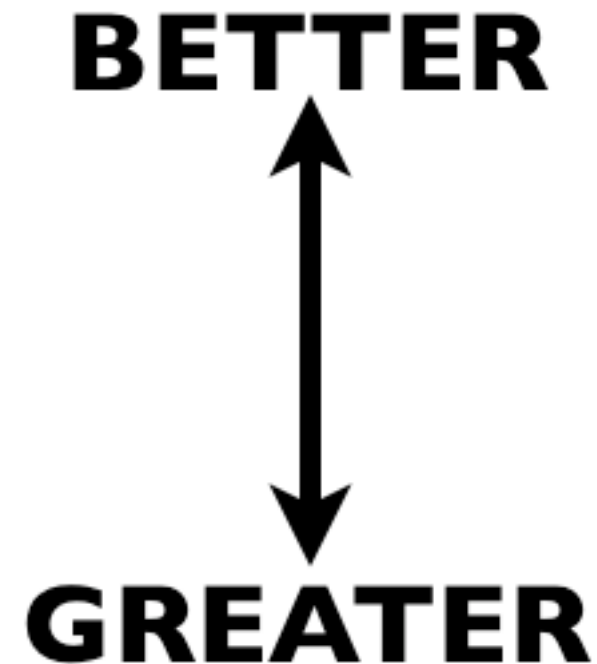
Diagram illustrating reuse distance calculation for an access trace: A, B, C, A, A, C, B. The reuse distance (rd) is shown below each element. The first three elements (A, B, C) have rd =  $\infty$  as they are the first occurrence of their respective data. The fourth element (A) has rd = 2, with a dotted oval labeled "use" above it. The fifth element (A) has rd = 0. The sixth element (C) has rd = 1, with a dotted oval labeled "reuse" above it. The seventh element (B) has rd = 2.



# Basic Idea

---

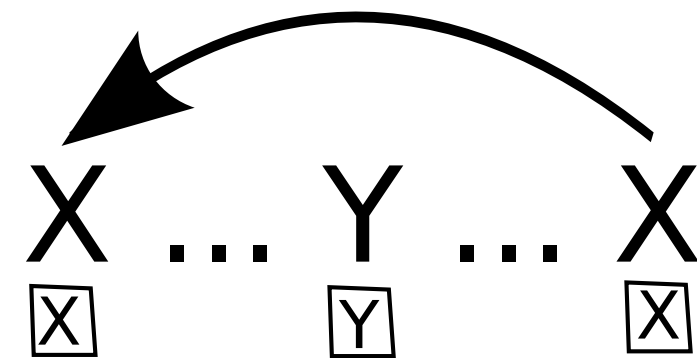
- Spatial locality
- Reduction on reuse distances when measuring block size doubled



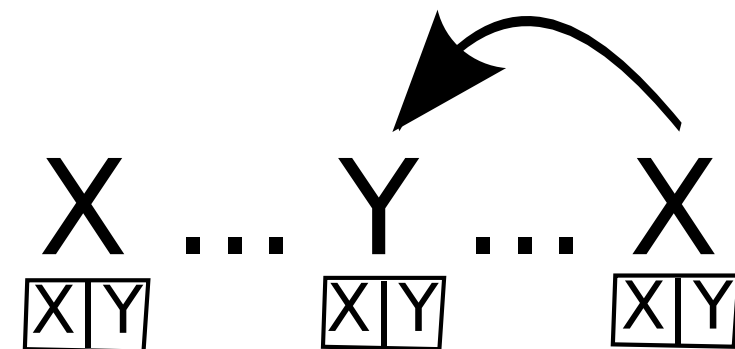
# Reduction in Reuse Distance

- Two cases of reductions
  - no intercept: reduced by half at most
  - intercept: may reduce it to constant


a long temporal reuse



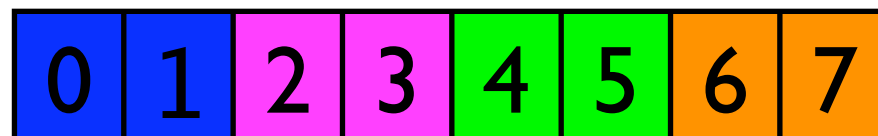
a short spatial reuse



# Contiguous Access

- An array with 8 elements 
  - access 0, 1, 2, 3, 4, 5, 6, 7 repeatedly
- Rd measured with single-element block
  - all are 7

- Rd measured with 2-element block



- half reduced to 0 (constant)
- half reduced to 3 (dependent on array size)

# Measuring Spatial Reuse

---

- Measure the change of every reuse distance
  - running two reuse distance analyzers simultaneously
- Criteria for effective spatial reuse
  - machine independent (reducing factor  $\geq 8$ )
  - machine dependent (L1&L2 cache sizes)



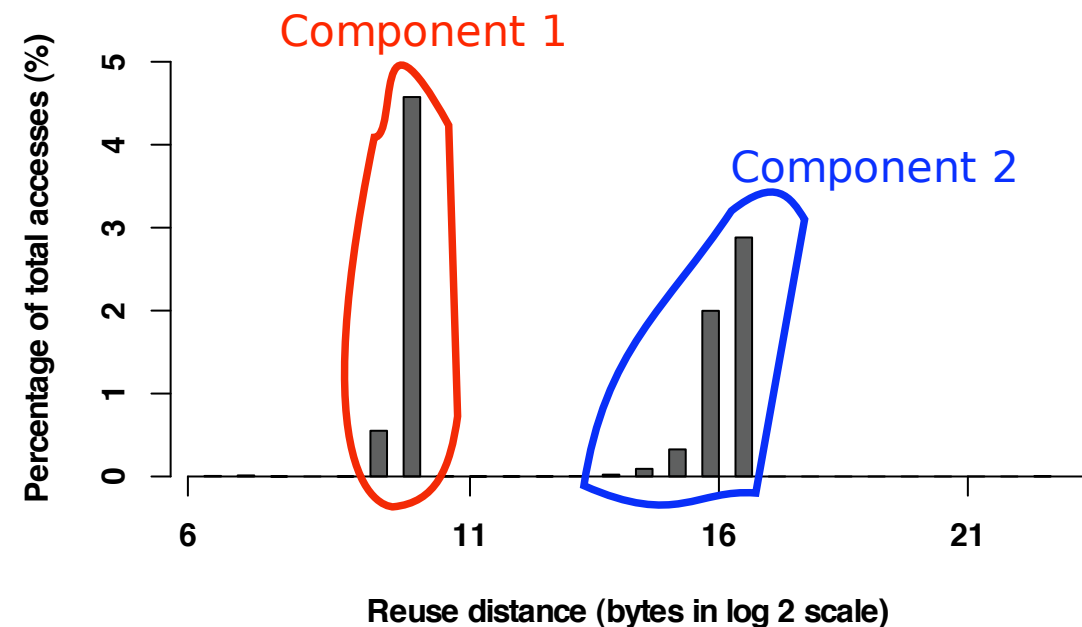
# Quantitative Scoring

---

- Spatial-locality quality score (SLQ)
  - portion of effective spatial reuse for each range of reuse distances, normalized to the best case (contiguous access)
  - 0 means no spatial locality and 1 means best spatial locality

# Locality Components

- Program level
- functions or loops
- Behavior level



- adjacent reuse distance ranges in length, which forms a peak
- similar in temporal locality but maybe very different in spatial locality

# Locality Components

---

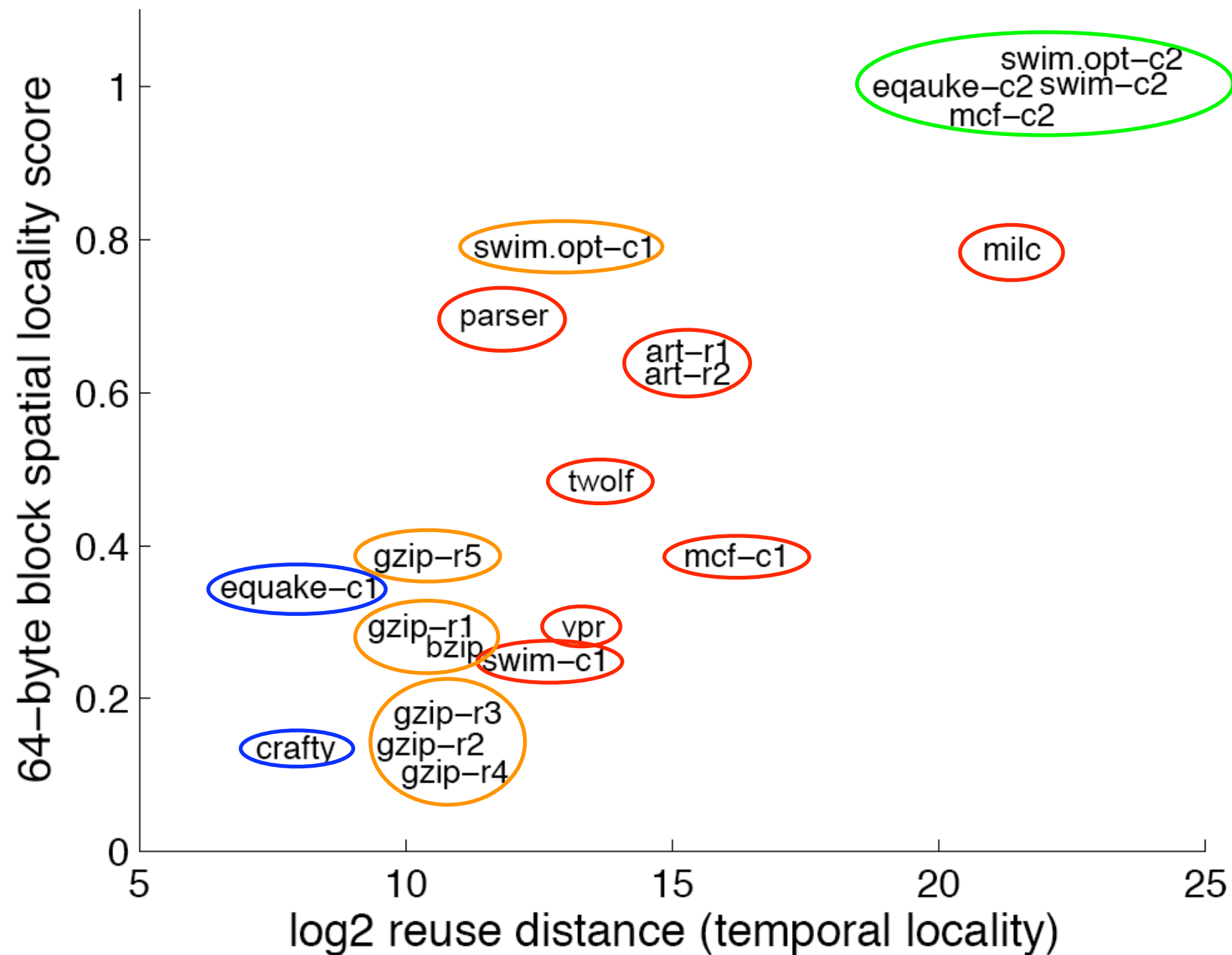
- Three metrics
  - total size
  - temporal locality: average reuse distance
  - spatial locality: average SLQ
- More findings than a single global result

# Locality Profiling Tools

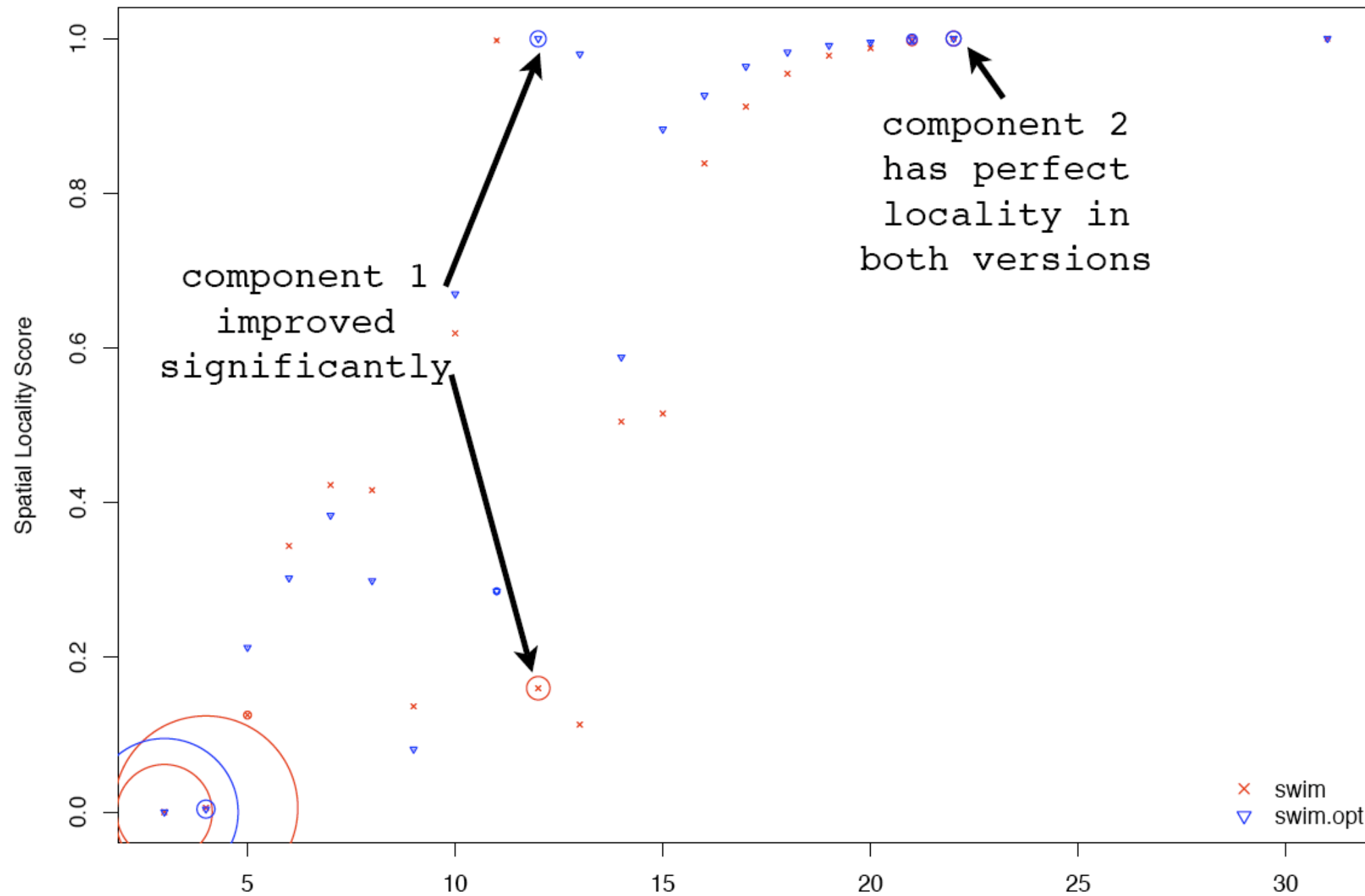
---

- Full trace analysis
  - dynamic binary instrumentation based on Valgrind
  - static source instrumentation based on GCC
- Sampling analysis
  - reservoir sampling (based on SLO [Beyls & D'Hollander])

# All Benchmark Results



# Result for Array Regrouping on Swim



# A User Study

---

- Locality tuning for a 2200-line machine translation application
  - using GCC-based tool with calling context tree
  - worst spatial locality occurs in a 10-line library function
  - 7% improvement after 6-line change
  - acceptable loss of precision in the translation

# Spatial Locality Ranking

- A sampling-based tool extended from SLO
- Worst spatial locality problems identified for mcf

The screenshot displays a software interface with two main panels. The left panel, titled 'Locality Problems' and 'Field Information', has tabs for 'Temporal', 'Block Temporal', and 'Spatial'. Under the 'Spatial' tab, a dropdown menu is set to 'L1 Raw'. The description reads: 'Poor Spatial Locality Reuse Rank List for Out of L1 Cache but In L2 Cache based on the raw number of bad accesses.' Below this, it shows 'L1 Cache Size(log2): 16' and 'L2 Cache Size(log2): 20', with 'Context Calling Tree On?: false'. A table lists the top 9 ranked items, with the first three highlighted in red. The right panel shows a snippet of C code with several expressions highlighted in red.

Rank	File	Function	Score	Percentage
1	mcfutil.c	refresh_potential	(7292/118124)	0.061732
2	mcfutil.c	refresh_potential	(4894/92248)	0.053053
3	pbla.c	primal_iminus	(1877/37842)	0.049601
4	pbla.c	primal_iminus	(1110/11740)	0.094549
5	treeup.c	update_tree	(926/31095)	0.029780
6	treeup.c	update_tree	(886/25245)	0.035096
7	pbla.c	primal_iminus	(654/8456)	0.077342
8	implicit.c	price_out_impl	(552/1608093)	0.000343
9	pbeampp.c	sort_basket	(96/194806)	0.000493

```
while( iplus != jplus )
{
  if( iplus->depth < jplus->depth )
  {
    if( iplus->orientation )
      TEST_MIN( iplus, 0, iplus->flow, > )
    else if( iplus->pred->pred )
      TEST_MIN( iplus, 0, (flow_t)1 - iplus->flow, > )
    iplus = iplus->pred;
  }
  else
  {
    if( !jplus->orientation )
      TEST_MIN( jplus, 1, jplus->flow, >= )
    else if( jplus->pred->pred )
      TEST_MIN( jplus, 1, (flow_t)1 - jplus->flow, >= )
    jplus = jplus->pred;
  }
}
```



# Summary

---

- A new model for spatial locality
  - based on reuse distance
  - models intra-block and adjacent-block spatial locality both
  - component-based, on both behavior and program levels
  - promising as part of performance tools

**Q & A**